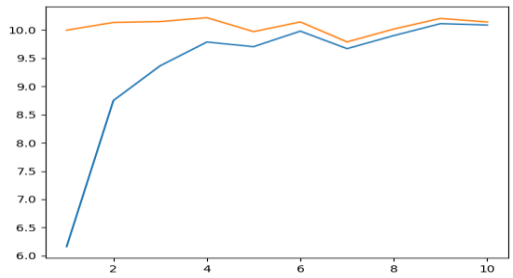
STAT 4360 (Introduction to Statistical Learning, Fall 2022)
Mini Project 2

Name: Haeun Kim

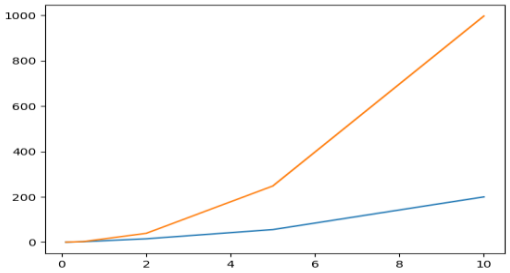
1-(a)

- Bias for JS = 15.143576596088838
- Bias for MLE = 28.398883706044014
- Risk for JS = 6.181823787912822
- Risk for MLE = 10.050927615155217

1-(b)

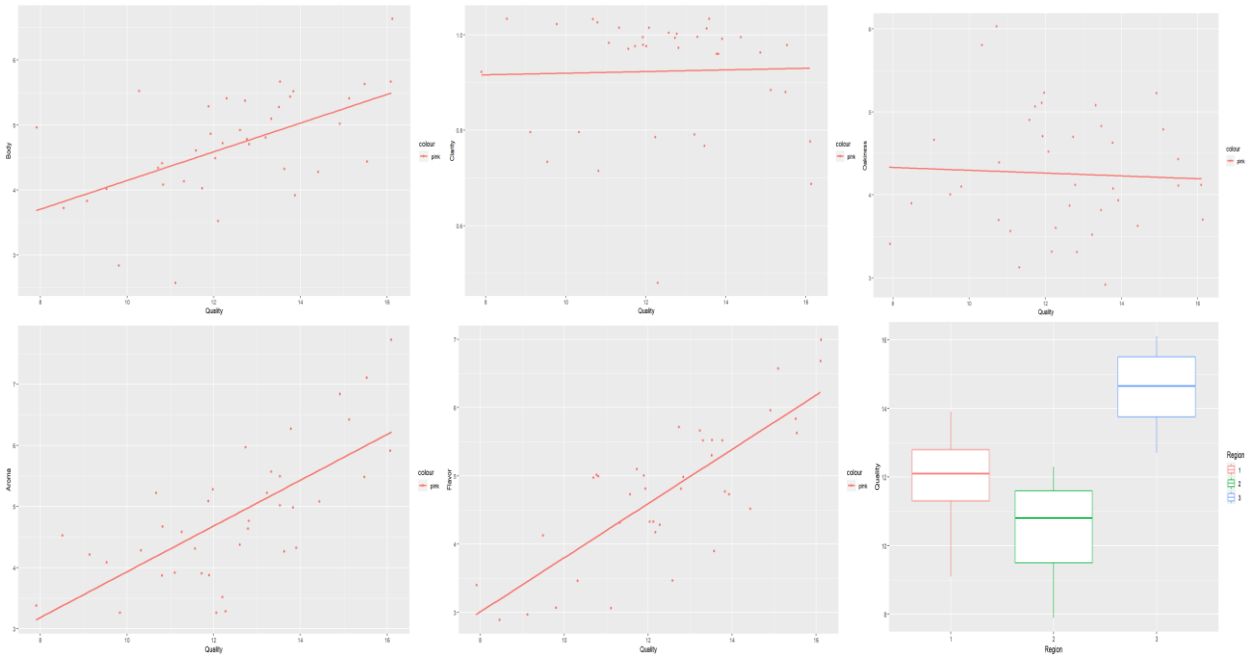


1-(c)



What I observed for 1-(b) is that as 'a' increase, risk of JS also increases. There are some part where risk decreases, but overally, the risk of JS increases. Especially, it dramatically increases when a is between 1 to 4. However, there's no big change for the risk of MLE as 'a' increase.So, I could conclude that coefficient of matrix does not effect on risk of MLE not too much.
What I observed for 1-(c) is that both of risk of JS and risk of MLE increase, as 'sigma' increase. Expecially, risk of MLE dramatically increases as sigma increase. However, the risk of JS gradually increases. So, I could think that the sigma affects on the risk of MLEIn both of plot, I also could observe the the risk of MLE is over the risk of JS.

2-(a)



- In order to analysis the data, I used Quality as response, and I tried to find out the relationship between Quality and other variables. First of all, Clarity and Oakiness have weak linear relationship with Quality. For Flavor, Aroma, and Body, I could think that those variables have the linear relationship with Quality. For Region, it has three regions. I could figure out that group 3's wine have the high quality, and group 1's wine have the second high quality, and group 2's wine show weak quality.

2-(b)

- I tries to fit a simple linear regression model to predict the response. And, I figured out that all variables except for Clarity and Oakiness have significant relationship to Quality. Because the p-values for Clarity and Oakiness is higher than p-value (0.05). The question says that I need to create some plots to back up my assertion, but I'm gonna use the plot from 2-(a). As I said above, there is not linear relationship between (Clarity and Oakiness) and Qaulity when we looked over the graph, while other variables have the linear relationship to the Quality. Surprisingly, the p-value (from simple linear regression model) supports my analysis.

2-(c)

- y (Qulity) = 7.81437 + 0.01705b1(Clarity) + 0.08901b2(Aroma) + 0.07967b3(Body) + 1.11723b4(Flavor) – 0.34644b5(Oakiness) – 1.51285b6(Region 2) + 0.97259b7(Region 3)

```
Call:
lm(formula = Quality ~ Clarity + Aroma + Body + Flavor + Oakiness +
    Region, data = wine)

Residuals:
     Min       1Q   Median       3Q      Max
-1.80824 -0.58413 -0.02081  0.48627  1.70909

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.81437    1.96944   3.968 0.000417
Clarity      0.01705    1.45627   0.012 0.990736
Aroma        0.08901    0.25250   0.353 0.726908
Body         0.07967    0.26772   0.298 0.768062
Flavor       1.11723    0.24026   4.650 6.25e-05
Oakiness    -0.34644    0.23301  -1.487 0.147503
Region2     -1.51285    0.39227  -3.857 0.000565
Region3      0.97259    0.51017   1.906 0.066218
```

When I looked over the summary(model), I could find that the Clarity, Aroma, Body, and Oakiness have high p-value. So, I think these does not have siginificant relationship with Quality to the full model. Predictors – Flavor and Region can be rejected the null hypothesis. That is, Flavor and Region have significant relationship to Quality

2-(d)

- I figured out that the flavor and region can be rejected the null hypothesis through P-values through above progress. So, I concluded that Flavor and Region have significant relationship to Quality. So I did AVONA with Full model and reduced model, which has only flavor and region as predictors. And I got the 0.6528 as the p-value for F-test statistic, and 0.6528 is less than $t_{\frac{a}{2},n-2}$. So, I could fail to reject the Null hypothesis. Therefore, I could get the reasonably good multiple linear regression.

2-(e)

Y = 7.094 + 1.116b4(Flavor) – 1.533(Region2) + 1.223(Region3)

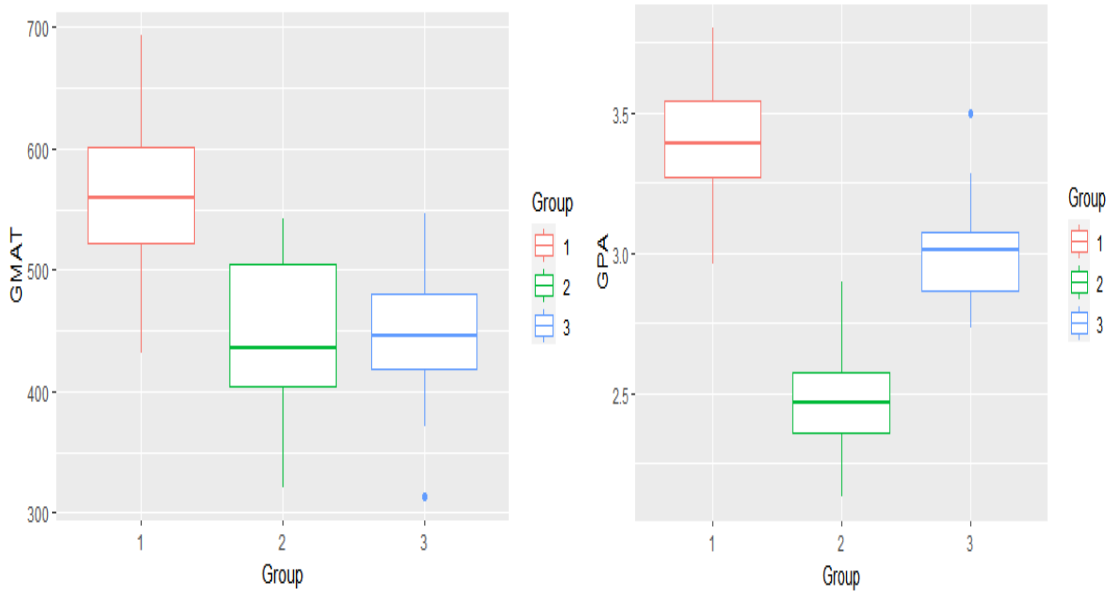
When the flavor increases by unit, y also increases by 1.116. 7.094 means the Quality of wine Region1. The region 3 makes the quality of wine increase by 1.223, while the region2 makes the quality of wine decrease by 1.533

2-(f)

- The prediction the Qaulity of a wine from Region 1 with means of Flavor is 12.41371. The 95% of prediction interval is [10.53775, 14.28967], and the 95% of confidence interval is [11.95152, 128759], is the mean of my estimate plus and minus the variation is that estimate.
Prediction interval is wider than the confidence interval because prediction interval depends on both the error from the fitted model and the error associated with future observations.

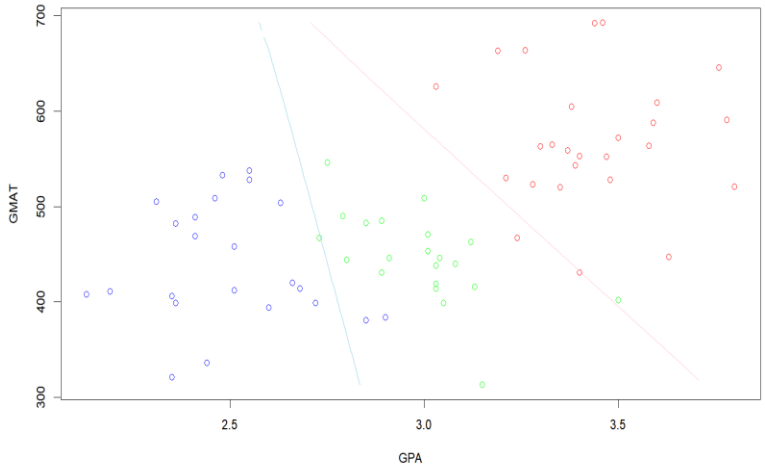
3-(a)

- There are three variables in thie dataset - GMAT, GPA, and Group. Group is response variables, and it is categorical data so, I thought boxplot is reasonable to analysis dataset. In case of GPA, it is well organized by group. This is because the boxes by group are not overlapped in terms of GPA. In case of GMAT, the Group 2 and Group 3 are a little ambiguous to analysis, but the Goup2 and Group3 are clearly distributed in regard to GPA.



3-(b)

```
Call:
lda(Group ~ GPA + GMAT, data = train)

Prior probabilities of groups:
        1         2         3
0.3714286 0.3285714 0.3000000

Group means:
       GPA     GMAT
1 3.431538 569.8077
2 2.496087 439.1304
3 2.990000 446.4286

Coefficients of linear discriminants:
              LD1         LD2
GPA  -5.103461571  2.07755357
GMAT -0.008900723 -0.01410918

Proportion of trace:
   LD1    LD2
0.9748 0.0252
```

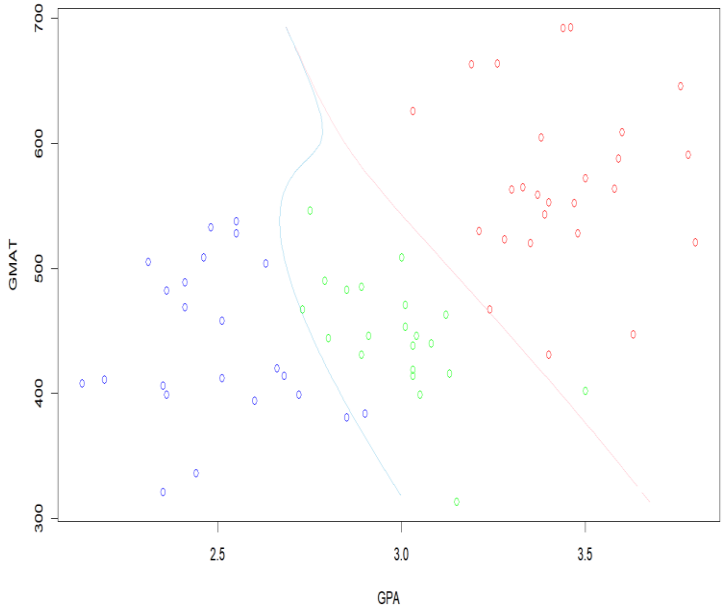


Training	1	2	3	Test	1	2	3
1	24	0	1	1	2	0	0
2	0	21	1	2	0	5	0
3	2	2	19	3	3	0	5

- I did perform LDA by using training data, and the result is above. I put the decision boundary, and I think the decision boundary seem sensible. It divides into three groups with little error. The two tables are Training and Test confusion matrixes respectively. And I figured out the misclassification rate of Training dataset is **0.08571429**, and the misclassification of Test dataset is **0.2**.

3-(c)

```
Call:
qda(Group ~ GPA + GMAT, data = train)

Prior probabilities of groups:
        1         2         3
0.3714286 0.3285714 0.3000000

Group means:
      GPA     GMAT
1 3.431538 569.8077
2 2.496087 439.1304
3 2.990000 446.4286
```



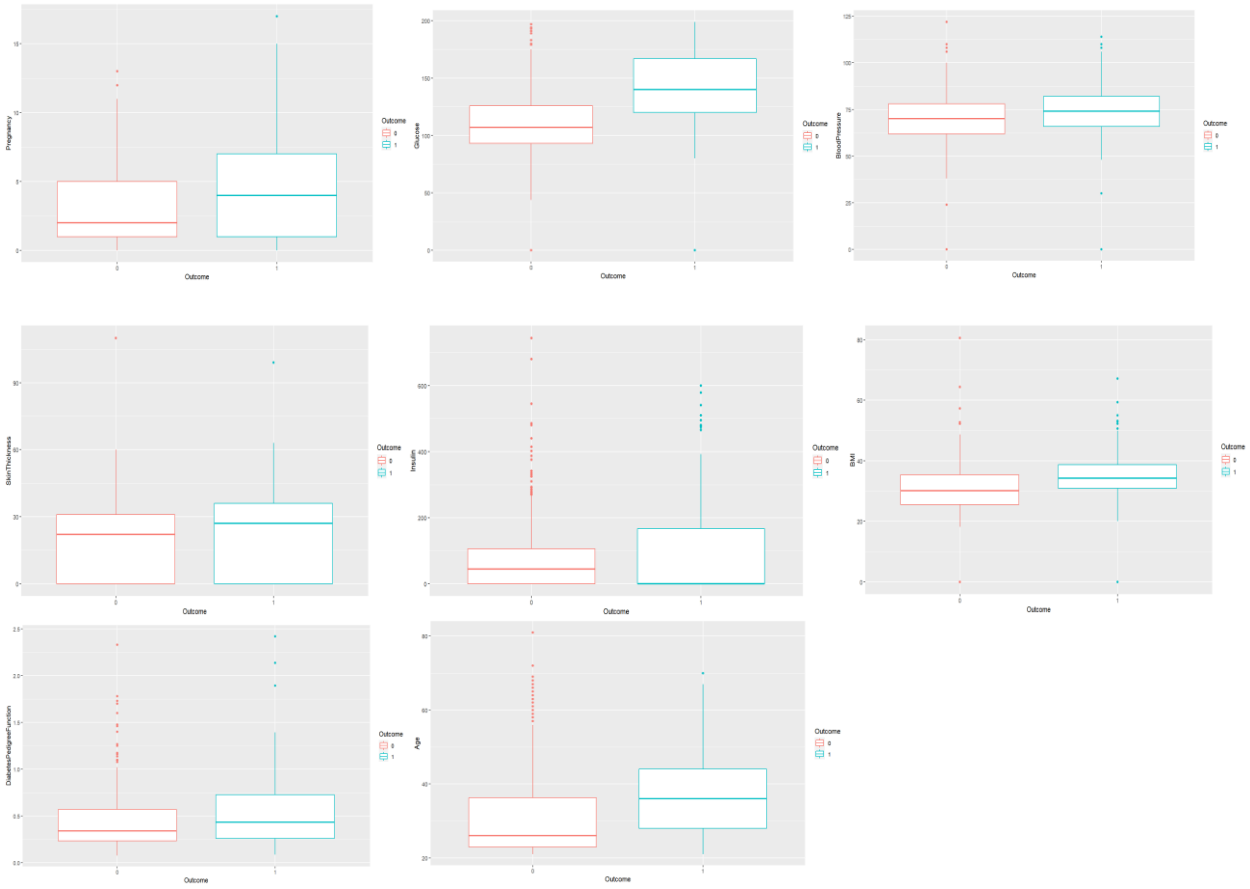
Training	1	2	3	Test	1	2	3
1	26	0	1	1	4	0	0
2	0	22	0	2	0	5	0
3	0	1	20	3	1	0	5

- I did perform QDA by using training data, and the result is above. I put the decision boundary, and I think the decision boundary seem sensible. It divides into three groups with little error. The two tables are Training and Test confusion matrixes respectively. And I figured out the misclassification rate of Training dataset is **0.02857143**, and the misclassification of Test dataset is **0.06666667**.

3-(d)

- I would recomment the QDA for this dataset. This is because the gap of miscalssification error for QDA between training dataset and test data is less than LDA. In addition, when I looked over the decision boundary from the plot, QDA has less error than LDA. So, for this dataset, I would recommend QDA instead of LDA.

4-(a)



- We set the Outcome as the reponse, the other variables as predictors. So, I want to figure out the relationship between response and predictors. And I used boxplot due to response is categorical. This is the boxplot for each variables versus Outcome. 0 means no diabetes, and 1 means diabetes from the Outcome. I could find that the all the predictors have some positive correlations with Outcome. Except for Insuline, all predictors' mean increases. In case of Insuline, although the mean decreases, but the 75th percentile increases, and maximum observation below upper fence also increases. So, I could think that all the predictors have some positive correlations with Outcome.

4-(b)

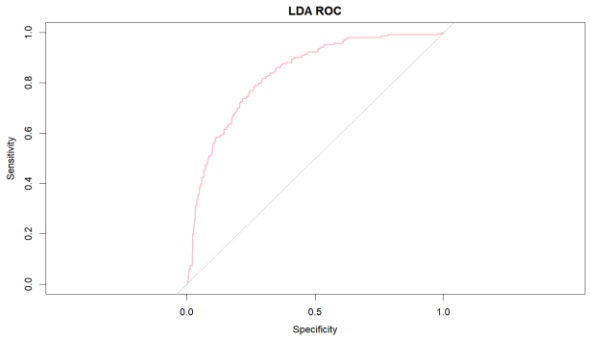
Performing an LDA of the data

```
Call:
lda(Outcome ~ Pregnancies.. + Glucose.. + BloodPressure.. + SkinThickness.. +
    Insulin.. + BMI.. + DiabetesPedigreeFunction.. + Age.., data = diabetes)

Prior probabilities of groups:
    0     1
0.658 0.342

Group means:
  Pregnancies.. Glucose.. BloodPressure..
0      3.168693  110.5866        68.09498
1      4.732456  141.5687        71.16667
  SkinThickness.. Insulin..    BMI..
0        20.05243  70.56383 30.56748
1        22.63304  98.89766 35.32047
  DiabetesPedigreeFunction..    Age..
0                  0.4346763 31.08131
1                  0.5406813 36.95614

Coefficients of linear discriminants:
                                     LD1
Pregnancies..                 0.1015317728
Glucose..                     0.0271248841
BloodPressure..              -0.0084340627
SkinThickness..               0.0012318356
Insulin..                    -0.0009708906
BMI..                         0.0537005742
DiabetesPedigreeFunction..    0.6549925988
Age..                         0.0109349308
```

Confusion Matrix

	0	1
0	1174	298
1	142	386

LDA Sensitivity
→ 0.5643275
LDA Specificity
→0.8920973
LDA Misclassification rate
→0.22



- Sensitivity is somewhat high, and specificity is also reasonable. So, LDA can be acceptable. The ideal ROC curve hugs the top left corner. In this LDA ROC curve, it is located upper left corner. So, this model shows good levels of both sensitivity and specificity.

4-(C)

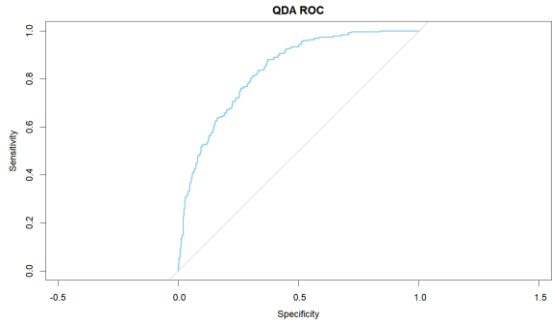
Performing an QDA of the data

```
Call:
qda(Outcome ~ Pregnancies.. + Glucose.. + BloodPressure.. + SkinThickness.. +
    Insulin.. + BMI.. + DiabetesPedigreeFunction.. + Age.., data = diabetes)

Prior probabilities of groups:
    0     1
0.658 0.342

Group means:
  Pregnancies.. Glucose.. BloodPressure.. SkinThickness.. Insulin..     BMI..
0      3.168693  110.5866         68.09498         20.05243  70.56383 30.56748
1      4.732456  141.5687         71.16667         22.63304  98.89766 35.32047
  DiabetesPedigreeFunction..     Age..
0                   0.4346763 31.08131
1                   0.5406813 36.95614
```

Confusion Matrix

	0	1
0	1135	290
1	181	394

QDA Sensitivity
→ 0.5760234
QDA Specificity
→ 0.862462
QDA Misclassification rate
→ 0.2355



- Sensitivity is somewhat high, and specificity is also reasonable. So, QDA can be acceptable. The ideal ROC curve hugs the top left corner. In this QDA ROC curve, it is located upper left corner. So, this model shows good levels of both sensitivity and specificity.

4-(D)

- There are no big differences between LDA and QDA in aspect of ROC. I would recomment LDA. This is because that the misclassification error for LDA is slight lower than QDA. For the recommended classifier, I suggest that the posterior probability cutoff is p = 0.2. I used laply function to figure out the cutoff from the ROC of LDA that I figured out above, and I could notice that when p = 0.1 or p = 0.2, the value has the highest among other cutoffs.

Python Code (or R Code)

```
## Question 1 ##
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from numpy.linalg import norm
#Basic necessary variables
p = 10
sgm = 1
mu = np.array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
mu_MLE = []
mu_JS = []
ip = np.identity(n=p)
n = 1000
#This function is for the bias for two estimators
def bias(mu_hat, mu):
   mu_hat = np.array(mu_hat)
   return norm(np.sum(mu_hat)/n - mu)

#This function is for the risk for two estimators
def risk(mu_hat, mu):
   mu_hat = np.array(mu_hat)
   return sum(norm(mu_hat_i - mu)**2
           for mu_hat_i in mu_hat
           )/n

  # return sum(norm(mu_hat_i - mu)**2 for mu_hat_i in mu_hat)/n
#This function is for the JAMES-STEIN Estimator
def james_stein(y, sigma):
   return (1-((p-2)*sigma**2)/norm(y)**2)*y


for y in np.random.multivariate_normal(mu, (sgm**2) * ip, n):
   mu_JS.append(james_stein(y, sgm))
   mu_MLE.append(y)
(a)
print(bias(mu_JS, mu))
print(bias(mu_MLE, mu))
print(risk(mu_JS, mu))
print(risk(mu_MLE, mu))
#Risk array for change of Sigma
risk_js2 = []
risk_mle2 = []
```

```
#Risk array for change a matrix
risk_jjs = []
risk_mlee = []

(b)
number = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# figuring out the multivariate normal distribution for the change of matrix
for a in number:
   mu_i = a * mu
   mu_JS1 = []
   mu_MLE1 = []
   for y in np.random.multivariate_normal(mu_i, (sgm**2)*ip, n):
      mu_JS1.append(james_stein(y, sgm))
      mu_MLE1.append(y)
   risk_jjs.append(risk(mu_JS1, mu_i))
   risk_mlee.append(risk(mu_MLE1, mu_i))

# Making the dataframe from what I found from previous progress
dff = pd.DataFrame({'number':number, 'risk_jjs':risk_jjs, 'risk_mlee':risk_mlee})
print(dff)
# Making the plot
plt.plot(number, risk_jjs)
plt.plot(number, risk_mlee)
plt.show()
(C)
ssigma = [0.1, 0.5, 2, 5, 10]
# figuring out the multivariate normal distribution for the change of sigma
for i in ssigma:
   mu_JS2 = []
   mu_MLE2 = []
   for y in np.random.multivariate_normal(mu, i**2 * ip, n):
      mu_JS2.append(james_stein(y, i))
      mu_MLE2.append(y)
   risk_js2.append(risk(mu_JS2, mu))
   risk_mle2.append(risk(mu_MLE2, mu))

# Making the dataframe from what I found from previous progress
df = pd.DataFrame({'ssigma':ssigma, 'risk_js2':risk_js2, 'risk_mle2':risk_mle2})
print(df)
# Making the plot
plt.plot(df['ssigma'], df['risk_js2'])
plt.plot(df['ssigma'], df['risk_mle2'])
plt.show()
```

```
library(class)
library(ISLR2)
library(MASS)
library(ggplot2)
library(pROC)
library(mvtnorm)
library(plyr)

setwd("C:/Users/haeun/OneDrive/문서/STAT33550")

#Problem 2
#Bring the wine data
wine <- read.table("wine.txt", header = TRUE)
#Looking over the brief structure of dataset
str(wine)
# Factoring the Region into 3
wine$Region = as.factor(wine$Region)
#(a)
ggplot(data = wine, aes(x = Region, y = Quality, color = Region)) +
  geom_boxplot()
ggplot(wine) +
  geom_jitter(aes(x = Quality, y = Clarity, color="pink")) +
  geom_smooth(aes(x = Quality, y = Clarity, color="pink"), method=lm, se=FALSE)
ggplot(wine) +
  geom_jitter(aes(x = Quality, y = Aroma, color="pink")) +
  geom_smooth(aes(x = Quality, y = Aroma, color="pink"), method=lm, se=FALSE)
ggplot(wine) +
  geom_jitter(aes(x = Quality, y = Body, color="pink")) +
  geom_smooth(aes(x = Quality, y = Body, color="pink"), method=lm, se=FALSE)
ggplot(wine) +
  geom_jitter(aes(x = Quality, y = Flavor, color="pink")) +
  geom_smooth(aes(x = Quality, y = Flavor, color="pink"), method=lm, se=FALSE)
ggplot(wine) +
  geom_jitter(aes(x = Quality, y = Oakiness, color="pink")) +
  geom_smooth(aes(x = Quality, y = Oakiness, color="pink"), method=lm, se=FALSE)


#(b)
# Simple linear regression between Quality to each variables
fit_clar <- lm(Quality ~ Clarity, data = wine)
fit_aroma <- lm(Quality ~ Aroma, data = wine)
fit_body <- lm(Quality ~ Body, data = wine)
fit_flavor <- lm(Quality ~ Flavor, data = wine)
```

```
fit_oakn <- lm(Quality ~ Oakiness, data = wine)
fit_region <- lm(Quality ~ Region, data = wine)
# To find the p-value, and the summarization of fit
summary(fit_aroma)
summary(fit_body)
summary(fit_clar)
summary(fit_flavor)
summary(fit_oakn)
summary(fit_region)
#(c)
#Full model for the dataset
full <- lm(Quality~Clarity + Aroma + Body + Flavor + Oakiness + Region, data = wine)
summary(full)
anova(full)
#(d)
full <- lm(Quality~Clarity + Aroma + Body + Flavor + Oakiness + Region, data = wine)
summary(full)
#Reduced model from the reasonably good linear regression
fit<-lm(Quality~Flavor + Region, data=wine)
anova(fit, full)
qf(0.95, 2, 30)
#(e)
# The final linear regression
fit
#(f)
newx <- data.frame(Flavor=mean(wine$Flavor),Region = as.factor(1))
 #95% prediction intervals
predict(fit, newx, interval= 'prediction')
 #95% confidence intervals
predict(fit, newx, interval='confidence')


####################################
#Problem 3
#Read the data
admission <- read.csv("admission.csv")
str(admission) # Group need to be factored
admission$Group <- as.factor(admission$Group) #factor the Group data

# Finding out the index for each Groups
group_1 <- subset(admission, Group == 1)
group_2 <- subset(admission, Group == 2)
group_3 <- subset(admission, Group == 3)
```

```
#For the group1, I made test dataset and train dataset
group_1_test <- group_1[1:5,]
group_1_train <- group_1[-c(1:5), ]
#For the group2, I made test dataset and train dataset
group_2_test <- group_2[1:5,]
group_2_train <- group_2[-c(1:5), ]
#For the group3, I made test dataset and train dataset
group_3_test <- group_3[1:5,]
group_3_train <- group_3[-c(1:5), ]

#This is test dataset
test <- rbind(group_1_test, group_2_test, group_3_test)
#This is training dataset
train <- rbind(group_1_train, group_2_train, group_3_train)
# X element for the training dataset
train.X <- train[,1:2]
# Y element for the training dataset
train.Y <- train[,3]
# X element for the test dataset
test.X <- test[,1:2]
# X element for the test dataset
test.Y <- test[,3]

#(a)
#ggplot for the distribution of GPA by group
ggplot(data = admission, aes(x = Group, y = GPA, color = Group))+
  geom_boxplot()
#ggplot for the distribution of GMAT by group
ggplot(data = admission, aes(x = Group, y = GMAT, color = Group))+
  geom_boxplot()
#(b)
# Perfoem an LDA using the training data
train_lda <- lda(Group ~ GPA + GMAT, data = train)
train_lda
train.pred.lda <- predict(train_lda, train)
#Confusion matrix for train dataset
table(train.pred.lda$class, train$Group)
# Calculate the error rate
training_error <- (1 + 1 + 2 + 2)/(24 + 1 + 21 + 1 + 2 + 2 + 19)
train.pred.test.lda <- predict(train_lda, test)
# Confusion matrix for test dataset
table(train.pred.test.lda$class, test$Group)
# Calculate the error rate
test_error <- (3)/(2 + 5 + 3 + 5)
```

```
s6340.lda <- function(y, X) {
  # y = training data response vector (a factor)
  # X = training data predictor matrix
  N <- length(y) # no of observations
  K <- nlevels(y) # no of classes
  p <- ncol(X) # no of predictors
  n <- as.numeric(table(y)) # class frequencies
  names(n) <- levels(y)
  pi <- n / N # class proportions
  # mean vector
  mu <- matrix(unlist(by(X, y, colMeans)), byrow = T, ncol = p)
  rownames(mu) <- levels(y)
  colnames(mu) <- colnames(X)
  # pooled covariance matrix
  S <- by(X, y, cov)
  Sigma <- Reduce("+", lapply(1:K, FUN = function(k) {
    (n[k] - 1) * S[[k]]
  })) / (N - K)
  # its inverse
  Sigma.inv <- solve(Sigma)
  # delta functions
  delta <- t(sapply(1:K, FUN = function(k) {
    c(-(1 / 2) * drop(t(mu[k, ]) %*% Sigma.inv %*% mu[k, ]) +
       log(pi[k]), t(mu[k, ]) %*% Sigma.inv)
  }))
  rownames(delta) <- levels(y)
  colnames(delta) <- c("(Intercept)", colnames(X))
  # pairwise difference of delta functions
  idx.pair <- combn(K, 2)
  delta.diff <- t(apply(idx.pair, MAR = 2, FUN = function(pair) {
    delta[pair[1], ] - delta[pair[2], ]
  }))
  rownames(delta.diff) <- apply(idx.pair, MAR = 2, FUN =
                                function(pair) {
                                  paste0(levels(y)[pair[1]], "-", levels(y)[pair[2]])
                                })
  # multiply intecept difference by 1 to get the cutoff c
  delta.diff[, 1] <- -delta.diff[, 1]
  colnames(delta.diff)[1] <- "Cutoff"
  # result
  result <- list(
    N = N, n = n, pi = pi, mu = mu, Sigma = Sigma,
    delta = delta, disc = delta.diff
```

```
  )
  return(result)
}
# by using s6340.lda function to figure out the lda
our.lda.fit <- s6340.lda(train.Y, train.X)
cutoff <- our.lda.fit$disc[1, 1]#Figuring out the cutoff
coeff <- our.lda.fit$disc[1, -1]#Figuring out the coeff
n.grid <- 50
# Make the grid for the dataset
x1.grid <- seq(f = min(train[, 1]), t = max(train[, 1]), l =
               n.grid)
x2.grid <- seq(f = min(train[, 2]), t = max(train[, 2]), l =
               n.grid)
grid <- expand.grid(x1.grid, x2.grid)
colnames(grid) <- colnames(train.X)
# prediction by lda
pred.grid <- predict(train_lda, grid)
prob1 <- matrix(pred.grid$posterior[, 1], nrow = n.grid, ncol = n.grid)
prob2 <- matrix(pred.grid$posterior[, 2], nrow = n.grid, ncol = n.grid)
colnames(grid) <- colnames(train.X)

# Making the plot for the LDA
plot(train.X, col =ifelse(train.Y == 1, "red", ifelse(train.Y == 2, "blue", "green")))
contour(x1.grid, x2.grid, prob1,
        levels = 0.5, labels = "", xlab = "", ylab = "",
        main = "", add = T, col = "pink")
contour(x1.grid, x2.grid, prob2,
        levels = 0.5, labels = "", xlab = "", ylab = "",
        main = "", add = T, col = "skyblue")
#(c)
# Perfoem an QDA using the training data
train_qda <- qda(Group ~ GPA + GMAT, data = train)
train_qda
train.pred.qda <- predict(train_qda, train)
#Confudsion matrix for train dataset
table(train.pred.qda$class, train$Group)
# Calculate error rate for train dataset
train_error <- (1 + 1)/ (26 + 1 + 22 + 1 + 20)
train.pred.test.qda <- predict(train_qda, test)
#Confusion matrix for test dataset
table(train.pred.test.qda$class, test$Group)
# Calculate error rate for test dataset
test_error <- (1)/(4 + 5 + 1 + 5)
#Superimpose the decision boundary
```

```
n.grid <- 50
x1.grid <- seq(f = min(train[, 1]), t = max(train[, 1]), l =
            n.grid)
x2.grid <- seq(f = min(train[, 2]), t = max(train[, 2]), l =
            n.grid)
grid <- expand.grid(x1.grid, x2.grid)
colnames(grid) <- colnames(train.X)
pred.grid.qda <- predict(train_qda, grid)
prob1 <- matrix(pred.grid.qda$posterior[, 1], nrow = n.grid, ncol = n.grid)
prob2 <- matrix(pred.grid.qda$posterior[, 2], nrow = n.grid, ncol = n.grid)

#Making the plot for QDA
plot(train.X, col =ifelse(train.Y == 1, "red", ifelse(train.Y == 2, "blue", "green")))
contour(x1.grid, x2.grid, prob1,
      levels = 0.5, labels = "", xlab = "", ylab = "",
      main = "", add = T, col = "pink")
contour(x1.grid, x2.grid, prob2,
      levels = 0.5, labels = "", xlab = "", ylab = "",
      main = "", add = T, col = "skyblue")

##########################################
#Problem 4
#Bring the data
diabetes <- read.csv("diabetes.csv")
#Factor the outcome (non-diabete, diabete)
diabetes$Outcome <- as.factor(diabetes$Outcome)
# make the data set for training dataset X
train.y <- diabetes$Outcome
# make the data set for training dataset X
train.x <- diabetes[,-9]
#(a)
# Make the boxplot to figure out the predictors to response(Outcome)
ggplot(data = diabetes, aes(x = Outcome, y = Pregnancies.., color =Outcome ))+
  geom_boxplot() +
  labs(x = "Outcome", y = "Pregnancy")
ggplot(data = diabetes, aes(x = Outcome, y = Glucose.., color = Outcome))+
  geom_boxplot()+
  labs(x = "Outcome", y = "Glucose")
ggplot(data = diabetes, aes(x = Outcome, y = BloodPressure.., color = Outcome))+
  geom_boxplot() +
  labs(x = "Outcome", y = "BloodPressure")
ggplot(data = diabetes, aes(x = Outcome, y = SkinThickness.., color = Outcome))+
  geom_boxplot()+
  labs(x = "Outcome", y = "SkinThickness")
```

```
ggplot(data = diabetes, aes(x = Outcome, y = Insulin.., color = Outcome))+
  geom_boxplot()+
  labs(x = "Outcome", y = "Insulin")
ggplot(data = diabetes, aes(x = Outcome, y = BMI.., color = Outcome))+
  geom_boxplot()+
  labs(x = "Outcome", y = "BMI")
ggplot(data = diabetes, aes(x = Outcome, y = DiabetesPedigreeFunction.., color = Outcome))+
  geom_boxplot()+
  labs(x = "Outcome", y = "DiabetesPedigreeFunction")
ggplot(data = diabetes, aes(x = Outcome, y = Age.., color = Outcome))+
  geom_boxplot()+
  labs(x = "Outcome", y = "Age")

#(b)
#Performing LDA for diabetes dataset
lda.dia <- lda(Outcome ~ Pregnancies.. + Glucose.. + BloodPressure.. + SkinThickness.. + Insulin..
+ BMI.. + DiabetesPedigreeFunction.. + Age.., data = diabetes)
lda.dia.pred <- predict(lda.dia, diabetes) # prediction from the LDA
table(lda.dia.pred$class, train.y) # confusion matrix
#Figuring out the ROC for LDA
roc.lda <- roc(train.y, lda.dia.pred$posterior[, "1"], levels = c("0", "1"))
#LDA'S sensitivity
lda.sen <- 386/(298+386)
#LDA'S specificity
lda.spe <- 1174/(1174+142)
#LDA's misclassification error rate
lda.error <- (298+142)/(1174+298+142+386)
#Making the plot for the ROC
plot(roc.lda, legacy.axes = T, col = "pink", xlab = "Specificity",
    ylab = "Sensitivity", main = "LDA ROC")

#(c)
#Performing QDA for diabetes dataset
qda.dia <- qda(Outcome ~ Pregnancies.. + Glucose.. + BloodPressure.. + SkinThickness.. +
Insulin.. + BMI.. + DiabetesPedigreeFunction.. + Age.., data = diabetes)
qda.dia.pred <- predict(qda.dia, diabetes) # Prediction from QDA
table(qda.dia.pred$class, train.y) #Confusion matrix
#Figuring out the ROC for QDA
roc.qda <- roc(train.y, qda.dia.pred$posterior[, "1"], levels = c("0", "1"))
# QDA's sensitivity
qda.sen <- 394/(290+394)
# QDA's specificity
qda.spe <- 1135/(1135+181)
# QDA's error rate
```

```
qda.error <- (290+181)/(1135+290+181+394)
# Making the plot for ROC
plot(roc.qda, legacy.axes = T, col = "skyblue", xlab = "Specificity",
    ylab = "Sensitivity", main = "QDA ROC")
#(4)
# To figure out the cutoff
cutoffs <- c(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9)
# Making the data frame from the ROC OF LDA
df <- with(roc.lda, data.frame(specificities, sensitivities, thresholds))
print(laply(cutoffs, function(cutoff) df$sensitivities[which.min(abs(df$specificities-cutoff))]))
```